จเร เลิศสุดวิชัย

ปรับแต่งแป้นพิมพ์

การเพิ่มขีดความสามารถของแป้นพิมพ์เพื่อนำไปประยุกต์ใช้งาน

มุมภาษาซีตอนที่แล้วได้กล่าวถึงการทำงานของแป้นพิมพ์ในเครื่องไมโครคอมพิวเตอร์ไปแล้วค่อนข้างละเอียด ในฉบับนี้ก็จะเป็นการปรับแต่งขีดความสามารถของแป้นพิมพ์นำไปประยุกต์ใช้ในโปรแกรมอื่นต่อไป

ปรับแต่งความเร็วแป้นพิมพ์

ปรับแต่งความเร็วของแป้นพิมพ์ในที่นี้คือ อัตราเร็วของการซ้ำแป้นพิมพ์ หมายถึง เมื่อเรากดปุ่มของแป้นพิมพ์ค้างเอาไว้ แป้นพิมพ์ของเครื่องจะสร้างตัวอักษรของปุ่มแป้นพิมพ์นั้นซ้ำออกมา ซึ่งอัตราเร็วของการสร้างตัวอักษรซ้ำนั้นล่ะคือ สิ่งที่เราจะปรับแต่งด้วยโปรแกรมภาษาซี

ก่อนอื่นมาพิจารณาความเป็นมาของอัตราเร็วของการซ้ำตัวอักษรกันก่อน ในเครื่องไมโครคอมพิวเตอร์รุ่นแรก คือ PC ผู้สร้างได้ออกแบบมาให้แป้นของเครื่องพีซีมีความสามารถในการซ้ำตัวอักษรอยู่แล้วด้วยอัตราเร็วที่สูงสุดของเครื่อง คือประมาณ 10 ตัวอักษรต่อวินาที ต่อมาในเครื่องรุ่น AT แป้นพิมพ์สามารถจะปรับเปลี่ยนอัตราเร็วการทำซ้ำตัวอักษรได้ โดยการเขียนค่าอัตราเร็วลงไปที่พอร์ตข้อมูลของแป้นพิมพ์ (KBDATA หรือหมายเลข 0×60) (การหาค่าอัตราเร็วซ้ำจะอธิบายต่อไป) เท่านั้นเอง จากนั้นต่อมาในปี ค.ศ. 1985 โปรแกรมไบออสที่ทำหน้าที่ควบคุมแป้นพิมพ์ก็ได้การปรับปรุงเพิ่มฟังก์ชันย่อยที่ใช้ในการปรับแต่งอัตราเร็วการซ้ำตัวอักษร คืออินเตอร์รัพต์ 0×16)

รูปที่ 1
โปรแกรมปรับเปลี่ยนอัตราเร็ว
การพิมพ์ซ้ำของแป้นพิมพ์

```
/* ------------- typematc.c -------------- */

/*
 * A program to reprogram the AT keyboard typematic rate
 */

#include <dos.h>
#include "interupt.h"

#define DELAY  0 /* delay factor after 1st keystroke:
                                        0,1,2,3 = 250,500,750,1000 m
iseconds */
#define REPEAT 0 /* repeat rate for keystrokes:
                        0 to 22 = 30 cps to 4 cps (approx)   */

/* ------- local prototypes ------- */
static void waitforkb(void);
static void sendkb(int);

void main(void)
{
        waitforkb();
        disable();
        waitforkb();
        enable();
        outportb(KBSTATUS, 0xad);
        waitforkb();
        sendkb(0xf3);
        sendkb((DELAY << 5) | REPEAT);
        waitforkb();
        outportb(KBSTATUS, 0xae);
}

static void waitforkb(void)
{
        int ct = 0;
        while (--ct)
                if ((inportb(KBSTATUS) & 2) == 0)
                        break;
}

static void sendkb(int c)
{
        int ct = 10000;
        outportb(KBDATA, c);
        while (--ct)
                ;
        outportb(KBSTATUS, 0xae);
}
```

ฟังก์ชัน 3 และสับฟังก์ชันที่ 5 (ซึ่งก็ไม่ได้มีอยู่ในทุกเครื่องเลยทีเดียว)

อัตราเร็วของการซ้ำตัวอักษร

มีค่าเวลาอยู่ 2 ค่าที่เกี่ยวข้องกับอัตราเร็วของการซ้ำตัวอักษร ค่าแรกคือค่าช่วงเวลาในการหน่วง เป็นช่วงเวลาที่นับจากการกดแป้นพิมพ์จนกระทั่งเริ่มการซ้ำตัวอักษร <--> อักษร ค่าเวลาหน่วงมีอยู่ด้วยกันสี่ค่าคือ 250 500 700 และ 1000 มิลลิวินาที

ตัวเลขค่าที่สองคืออัตราเร็วของการซ้ำตัวอักษรมีหน่วยเป็นตัวอักษรต่อวินาที ซึ่งมีอยู่ด้วยกัน 31 ค่าที่เป็นไปได้ โดยเริ่มจากอัตราการซ้ำที่ช้าสุดประมาณ 2 ตัวอักษรต่อวินาที จนถึงเร็วสุดคือประมาณ 30 ตัวอักษรต่อวินาที

โปรแกรมปรับเปลี่ยนอัตราเร็วการซ้ำ

จากที่กล่าวมาแล้วสำหรับเครื่อง AT สามารถที่ปรับเปลี่ยนอัตราซ้ำของแป้นพิมพ์ได้ ซึ่งในโปรแกรมรูปที่ 1 โปรแกรม Typemate.c เป็นโปรแกรมที่ทำหน้าที่ปรับเปลี่ยนอัตราเร็วการพิมพ์ซ้ำของแป้นพิมพ์ ถ้าสังเกตโดยจะพบการปรับเปลี่ยนจะทำโดยการนำเอาค่าจากตัวคงที่ที่กำหนดเอาไว้สองค่าคือ DELAY และ REPEAT ออกไปที่พอร์ต KBDATA (พอร์ตหมายเลข 0×60) ประกาศอยู่ในไฟล์ Interrupt.h ซึ่งลงไปแล้วในตอนก่อน) ตามขั้นตอนของการปรับเปลี่ยนค่าของแป้นพิมพ์ (พิจารณาโปรแกรมตามไปด้วย) สาเหตุที่ทำการปรับเปลี่ยนค่าโดยตรงที่พอร์ตแทนการเรียกใช้ฟังก์ชันหรืออินเตอร์รัพต์บริการของไบออสก็เนื่องจากเครื่อง AT ไม่ได้มีฟังก์ชันบริการในการปรับเปลี่ยนอัตรา

ซ้ำของแป้นพิมพ์ทุกเครื่อง ขึ้นอยู่กับรุ่นของไบออสด้วยเช่นกัน โดยเฉพาะไบออสในรุ่นต่อ ๆ มา

การปรับเปลี่ยนอัตราเร็วการพิมพ์ของเครื่อง PC และ XT

ดังได้กล่าวมาแล้วว่าในเครื่อง PC และ XT ซึ่งเป็นเครื่องรุ่นเก่า อัตราเร็วของการพิมพ์จะถูกกำหนดตายตัวมาพร้อมกับเครื่อง แล้วเราสามารถปรับเปลี่ยนอัตราเร็วดังกล่าวได้หรือไม่อย่างไร ?

คำตอบก็คือ สามารถทำได้ โดยการเขียนโปรแกรมในลักษณะของการตรวจจับการกดแป้นพิมพ์ ซึ่งก่อนอื่นมาลองย้อนไปดูเรื่องของแป้นพิมพ์ในกรณีที่มีการกดและปล่อยแป้นพิมพ์กันก่อนดังนี้

เมื่อมีการกดแป้นพิมพ์ จะเกิดอินเตอร์รัพต์หมายเลข 9 ขึ้น ซึ่งโปรแกรมอินเตอร์รัพต์คือไบออสส่วนที่เกี่ยวกับแป้นพิมพ์จะไปอ่านข้อมูลจากพอร์ตข้อมูลของแป้นพิมพ์ (0×16) และเมื่อมีการปล่อยแป้นพิมพ์ที่ถูกก็จะเกิดอินเตอร์รัพต์ขึ้นเช่นเดิม และคราวนี้ค่าที่อ่านได้จากพอร์ตข้อมูลของแป้นพิมพ์จะยังคงเป็นค่าเดิม แต่บิตซ้ายสุดของค่ารหัสสแกนที่อ่านได้จะถูกถอดให้เป็น 1 ซึ่งจากลักษณะการทำงานแบบนี้ของแป้นพิมพ์สามารถนำมาใช้ในการเขียนโปรแกรมที่สามารถปรับเปลี่ยนอัตราเร็วการพิมพ์ซ้ำได้

จะเห็นว่าโปรแกรมที่สร้างต้องคอยตรวจสอบการกดและปล่อยแป้นพิมพ์ตลอดเวลา รวมทั้งต้องคอยสร้างตัวอักษรที่ซ้ำด้วย ดังนั้นโปรแกรมในรูปที่ 2

รูปที่ 2 โปรแกรม Fastkeys.c

```
/* ---------- fastkeys.c ---------- */

#include <dos.h>
#include <stdlib.h>
#include "interupt.h"
#include "keys.h"

#define TICKS 6

static int lastscan = 0;
static int stuffing = 0;
static int keyrate = 1;
static int ticker = TICKS;

static void (interrupt far *oldtimer)(void);
static void interrupt (*oldkb)(void);
static void interrupt (*oldkeyboard)(void);

static void interrupt far newtimer(void);
static void interrupt newkb(void);
static void interrupt newkeyboard(IREGS);

/* ------- timer interrupt service routine ------- */
static void interrupt newtimer(void)
{
        static int i;

        if (lastscan && stuffing)
                if (ticker)
                        --ticker;
                else
                        for (i = 0; i < keyrate; i++)
                                stuffkey(lastscan);
    (*oldtimer)();
}

static void interrupt newkb(void)
{
        static int kbfunc, i;
        static int nullkeys[] = {56,58,42,54,29,69,70,84};

        if ((kbfunc = inportb(KBDATA)) & 128)
                stuffing = 0, ticker = TICKS, lastscan = 9;
        else    {
                for (i = 0; i < 8; i++)
                        if (kbfunc == nullkeys[i])
                                break;
                stuffing = (i == 8);
        }
        (*oldkb)();
}

static void interrupt newkeyboard(IREGS ir)
{
        static kbfunc;

        kbfunc = (ir.ax >> 8) & 255;
        _CX = ir.cx;
        _BX = ir.bx;
        _AX = ir.ax;
        (*oldkeyboard)();
        ir.ax = _AX;
        ir.fl = _FLAGS;
        if (kbfunc == 0)
                lastscan = ir.ax;
}

void main(int argc, char *argv[])
{
        if (argc > 1)
                keyrate = atoi(argv[1]);
    oldtimer = getvect(TIMER);
    setvect(TIMER, newtimer);
    oldkeyboard = getvect(KEYBOARD);
    setvect(KEYBOARD, newkeyboard);
    oldkb = getvect(KYBRD);
    setvect(KYBRD, newkb);
        keep(0, 200);
}
```

(โปรแกรม Fastkey.c) โดยโปรแกรมในรูปที่ 2 Fastkey.c สามารถทำงานได้ทั้งในเครื่องพีซี XT รวมทั้งเครื่อง AT ที่มีและไม่มีคุณสมบัติพิเศษในการปรับเปลี่ยนอัตราเร็วการพิมพ์ซ้ำ แต่ถ้าเป็นเครื่องที่สามารถปรับได้ไม่ควรจะใช้โปรแกรมนี้เพราะเปลืองหน่วยความจำไม่สมควร

การใช้งานโปรแกรมนี้ก็เหมือนกับโปรแกรมเรสซิเดนต์ทั่วไปคือ จะต้องเรียกโปรแกรมนี้ที่ดอสแล้วตามด้วยพารามิเตอร์ที่ระบุอัตราเร็วการพิมพ์ซ้ำ ซึ่งตามปกติจะกำหนดมีค่าเป็น 1 อยู่แล้ว โดยจะทำให้เกิดอัตราเร็วการซ้ำประมาณ 26 ตัวอักษรต่อวินาที ซึ่งใกล้เคียงกับอัตราเร็วสูงสุดของเครื่อง AT (30 ตัวอักษรต่อวินาที) ควรระวังค่าอัตราเร็วที่มากเกินไปจะทำให้ไม่สามารถควบคุมการกดแป้นพิมพ์ได้ ซึ่งค่าประมาณ 3 ก็มากเกินไปแล้ว อย่างไรก็ตาม ขึ้นอยู่กับโปรแกรมและการใช้งานของผู้ใช้โปรแกรม

ต่อไปมาพิจารณาการทำงานของโปรแกรมกันสักเล็กน้อย เมื่อเรียกโปรแกรม Fastkey จะเปลี่ยเวกเตอร์อินเตอร์รัพต์ของเวลา (timer หมายเลข 8) เวกเตอร์อินเตอร์รัพต์ฮาร์ดแวร์แป้นพิมพ์ (KYBRD หมายเลข 9) และเวกเตอร์อินเตอร์รัพต์บริการแป้นพิมพ์ของไบออส (KEYBOARD หมายเลข 0×16) จากนั้นจะฝังโปรแกรมลงในหน่วยความจำ เมื่อมีการกดแป้นพิมพ์ ฟังก์ชัน newkb ซึ่งแทนที่โปรแกรมย่อยอินเตอร์รัพต์หมายเลข 9 เดิมจะทำงานโดยอ่านค่ารหัสสแกนจากพอร์ตข้อมูลของแป้นพิมพ์ (พอร์ตหมายเลข 0×61) KBDATA) โดยตรวจสอบบิตซ้ายสุดของข้อมูลดังกล่าว ว่าเป็นการกดแป้นพิมพ์หรือเป็นการปล่อยแป้นพิมพ์ ถ้าหากว่าเป็นการกดแป้นพิมพ์จะทำการเซตแพล็ก stuffing บอกฟังก์ชัน newtimer ให้เริ่มต้นการพิมพ์ซ้ำโดยในฟังก์ชัน newtimer ซึ่งแทนที่โปรแกรมอินเตอร์รัพต์หมายเลข 8 เดิมก่อนจะเริ่มซ้ำตัวอักษรจะหน่วงเวลาก่อนโดยใช้ค่าในตัวแปร ticker เป็นค่าหน่วงเวลา จากนั้นทุกครั้งที่เกิดอินเตอร์รัพต์เวลา (ประมาณ 18.2 ครั้งต่อวินาที) ฟังก์ชัน newtimer จะนำเอาค่ารหัสสแกนที่เกิดจากแป้นพิมพ์ที่กดลงเก็บในบัฟเฟอร์แป้นพิมพ์โดยใช้ฟังก์ชัน stuffkey (จากโปรแกรมของมุมภาษาซีตอนที่แล้ว)

และเมื่อปล่อยแป้นพิมพ์ (ตรวจสอบจากข้อมูลที่อ่านจากพอร์ต 0×61) บิตซ้ายจะเป็นหนึ่ง) ฟังก์ชัน newkb จะเคลียร์แพล็ก stuffing เพื่อบอกฟังก์ชัน newtimer เลิกซ้ำตัวอักษร อย่างไรก็ตามมีแป้นพิมพ์บางคีย์ที่โปรแกรมไบออสฮาร์ดแวร์ แป้นพิมพ์เดิมไม่สร้างรหัส แต่จะไปเก็บค่าที่หน่วยความจำสถานะแป้นพิมพ์คือที่ตำแหน่งหน่วยความจำ 0×471 แทนคีย์เหล่านั้น ได้แก่ แป้นพิมพ์ Ctrl, Alt, Shift, Caps Lock, Num Lock, Scroll Lock และ Break จะไปทำงานอะไรเมื่อตรวจสอบแล้วพบว่าการกดแป้นพิมพ์เป็นคีย์พิเศษดังกล่าว

สำหรับฟังก์ชัน newkeyboard ซึ่งทำงานแทนโปรแกรมบริการไบออสแป้นพิมพ์นั้นจะทำหน้าที่ส่วนหนึ่งในการบอกการเริ่มต้นการซ้ำแป้นพิมพ์ด้วย เช่นซึ่งการซ้ำแป้นพิมพ์จะเริ่มต้นเมื่อริ่มมีการกดแป้นพิมพ์และมีโปรแกรมประยุกต์ต้องการอ่านแป้นพิมพ์ ซึ่งฟังก์ชัน newkeyboard จะจำค่ารหัสสแกนแป้นพิมพ์เอาไว้ในตัวแปร lastscan โดยฟังก์ชัน newtimer จะนำค่า lastscan ดังกล่าวไปใช้ในการซ้ำแป้นพิมพ์

อัตราเร็วของการซ้ำจะคำนวณอย่างคร่าว ๆ ได้ดังนี้คือ อัตราเร็วการซ้ำแป้นพิมพ์ตามปกติของแป้นพิมพ์พีซี XT คือประมาณ 10 ตัวอักษรต่อวินาที รวมกับอัตราเร็วที่กำหนดให้โปรแกรม Fastkeys โดยนำเร่งค่าที่ใช้เป็นพารามิเตอร์ตอนเรียกโปรแกรมคูณด้วย 18 เพราะเนื่องจากอินเตอร์รัพต์เวลาเกิดขึ้น

ทุก 18.2 ครั้ง ซึ่งโดยสรุปอัตราเร็วของการซ้ำการพิมพ์ที่ได้จากโปรแกรม Fastkeys จะมีค่าประมาณ 10 + 18*P โดย P คือค่าที่ใช้กำหนดอัตราเร็วตอนเรียกโปรแกรม อย่างไรก็ตาม อัตราเร็วดังกล่าวเป็นการคำนวณแต่ในทางปฏิบัติอัตราเร็วของการพิมพ์ซ้ำจะได้รับผลจากเหตุการณ์อื่น ๆ

แป้นพิมพ์ที่พิมพ์เองอัตโนมัติ

ในการใช้งานโปรแกรมคอมพิวเตอร์มีบ่อยครั้งที่เดียวที่ผู้ใช้โปรแกรมต้องพิมพ์ข้อความหรือทำงานบางอย่างที่ซ้ำ ๆ กัน ซึ่งหากสามารถทำให้เครื่องคอมพิวเตอร์จดจำข้อความ หรือการกระทำเหล่านั้นแล้วนำมากระทำซ้ำให้ก็จะช่วยอำนวยความสะดวกให้กับผู้ใช้โปรแกรมมากทีเดียว ในปัจจุบันมีโปรแกรมประยุกต์หลายโปรแกรมที่มีคุณสมบัติดังกล่าว ซึ่งเรียกคุณสมบัตินี้ว่า คีย์บอร์ดมาโคร ตัวอย่างเช่น โปรแกรม SuperKey และโปรแกรม ProKey โปรแกรมเหล่านี้สามารถจะจดจำข้อความหรือชุดลำดับการกดแป้นพิมพ์แล้วแทนด้วยการกดแป้นพิมพ์ใดแป้นพิมพ์หนึ่งตามแต่จะเลือก โปรแกรมดังกล่าวก็จะพิมพ์ข้อความที่มันจำออกมาให้คุณ คุณสมบัตินี้ปัจจุบันยังพบในโปรแกรมเอดิเตอร์และเวิร์ดโปรเซสซิงด้วย

ลักษณะการพิมพ์อัตโนมัตินี้ยังมีการนำไปใช้ในการสร้างตัวอย่างโปรแกรมโดยผู้พัฒนาโปรแกรมต้องการแสดงให้ผู้ใช้ได้เห็นถึงขั้นตอนการทำงานของโปรแกรม ซึ่งจะมีการทำงานเองอัตโนมัติเหมือนกับมีคนกำลังใช้งานโปรแกรมนั้นอยู่ ลักษณะตัวอย่างโปรแกรมเหล่านี้มักจะพบในการโฆษณาขายโปรแกรม หรือพบในกรณีโปรแกรมช่วยสอนการใช้งาน (Futorial program)

โปรแกรมช่วยพิมพ์อัตโนมัติ

ถ้าหากยังจำได้ถึงมุมภาษาซีในตอนที่แล้ว ซึ่งมีโปรแกรมช่วยนำรหัสแกนไปใส่ไว้ในบัฟเฟอร์แป้นพิมพ์เสมือนกับมีการพิมพ์จากแป้นพิมพ์โดยตรง การสร้างโปรแกรมช่วยพิมพ์อัตโนมัติก็จะทำได้จากวิธีการดังกล่าว

รูปที่ 3 โปรแกรม Autotype

```
1  /* --------- autotype.c ---------- */
2
3  /*
4   * Automatic typing. Stuff a string of characters into the
5   * BIOS keyboard read-ahead buffer.
6   */
7
8  #include <stdio.h>
9  #include <dos.h>
10 #include <string.h>
11 #include "interupt.h"
12 #include "keys.h"
13
14 static char biosbuffer[256];
15 static char *nxton  = biosbuffer;
16 static char *nxtoff = biosbuffer;
17 static int rate = 0;
18 static void interrupt (*oldkb)(void) = NULL;
19 static void interrupt newkb(IREGS);
20 static void insertkey(int);
21 static void nextbuff(void);
22
23 /* ----- autotype a string as if from the keyboard ----- */
24 void autotype(char *s)
25 {
26         while (*s)
27                 insertkey(*s++);
28         if (oldkb == NULL)      {
29                 oldkb = getvect(KEYBOARD);
30                 setvect(KEYBOARD, newkb);
31         }
32 }
33
34 /* ----- set repeat rate in timer ticks ------------ */
35 void repeat_rate(int n)
36 {
37         rate = n;
38 }
39
40 /* ----- put a keystroke into the autotype buffer ------ */
41 static void insertkey(int c)
42 {
43         if (nxton+1 == nxtoff ||
44                 (nxton+1 == biosbuffer+256 && nxtoff == biosbuff
   )
45                 return;
46         *nxton++ = c;
47         if (nxton == biosbuffer+256)
48                 nxton = biosbuffer;
49 }
50
51 static void interrupt newkb(IREGS ir)
52 {
53         static kbfunc;
54
55         kbfunc = (ir.ax >> 8) & 255;
56         if (kbfunc < 2 && nxton != nxtoff)      {
57                 /* --- read or status check w/ chars in buffer -
   */
58                 if (kbfunc == 1)        {
59                         /* ----- status check ------ */
60                         if (*nxtoff == -1)      {
61                                 /* ----- force no character ready
   --- */
62                                 ir.fl |= ZEROFLAG;
63                                 nextbuff();
```

รูปที่ 3 (ต่อ)

```
64                              }
65                              else    {
66                                      /* --- return character, don't a
    nce --- */
67                                      ir.ax = (*nxtoff) & 0xff;
68                                      ir.fl &= ~ZEROFLAG;
69                              }
70                              return;
71                      }
72                      /* ------- read keyboard -------- */
73                      while (*nxtoff == -1 && nxtoff != nxton)
74                              nextbuff();
75                      if (nxtoff != nxton)    {
76                              if (*nxtoff > 127)      {
77                                      ir.ax = 0;
78                                      *nxtoff &= 0x7f;
79                              }
80                              else    {
81                                      ir.ax = (*nxtoff) & 0xff;
82                                      nextbuff();
83                                      enable();
84                                      delay(rate);
85                                      disable();
86                              }
87                              return;
88                      }
89              }
90              /* ---- not read or status or buffer is empty ---- */
91              _CX = ir.cx;
92              _BX = ir.bx;
93              _AX = ir.ax;
94              (*oldkb)();
95              ir.ax = _AX;
96              ir.fl = _FLAGS;
97      }
98
99      /* ------ increment next off buffer pointer ------- */
100     static void nextbuff(void)
101     {
102             nxtoff++;
103             if (nxtoff == biosbuffer+256)
104                     nxtoff = biosbuffer;
105             if (nxtoff == nxton)    {
106                     setvect(KEYBOARD, oldkb);
107                     oldkb = NULL;
108             }
109     }
```

นั่นเอง แต่ปัญหาที่จะพบก็คือบัฟเฟอร์ของแป้นพิมพ์จริง ๆ มีที่เก็บอยู่เพียง 15 คีย์เท่านั้น ซึ่งเป็นจำนวนที่น้อยเกินไปต่อการใช้งาน

ในโปรแกรมรูปที่ 3 โปรแกรม autotype.c เป็นโมดูลที่มีฟังก์ชันสำหรับช่วยพิมพ์อัตโนมัติได้ โดยสามารถพิมพ์ข้อความล่วงหน้าได้ถึง 256 ตัวอักษร ในเรียกฟังก์ชัน autotype พร้อมพารามิเตอร์คือพอยน์เตอร์ของข้อความที่ต้องการ ภายในข้อความสามารถจะบรรจุค่ารหัสของแป้นพิมพ์พิเศษต่าง ๆ ได้ (ตามค่าในไฟล์ Keys.h ในตอนที่แล้ว) นอกเหนือจากตัวอักษรปกติธรรมดา การที่สามารถพิมพ์ข้อความได้ถึง 256 ตัวก็เพราะว่าโปรแกรมได้เปลี่ยนที่ของบัฟเฟอร์แป้นพิมพ์ออกมาเป็นบัฟเฟอร์วงกลมที่สร้างขึ้นเองในโปรแกรมคือ bios buffer

ก่อนที่จะเรียกฟังก์ชัน autotype สามารถปรับแต่งอัตราเร็วของการพิมพ์ข้อความได้ด้วยฟังก์ชัน repeat_rate โดยระบุเป็นจำนวนมิลลิวินาที ซึ่งในลักษณะของโปรแกรมตัวอย่างหรือโปรแกรมช่วยสอนควรจะทำงานช้า เพื่อผู้ใช้จะได้สังเกตการทำงานโปรแกรมได้ทัน

การทำงานของ autotype นั้น เมื่อถูกเรียกครั้งแรกก็จะเปลี่ยนอินเตอร์รัพต์เวกเตอร์หมายเลข 0×16 (KEYBOARD) โดยแทนที่ด้วยฟังก์ชัน newkb ทุกครั้งที่โปรแกรมประยุกต์อ่านแป้นพิมพ์ ฟังก์ชัน newkb จะทำงานแทนโปรแกรมบริการแป้นพิมพ์ของไบออสแทน โดยจะอ่านค่าแป้นพิมพ์จากบัฟเฟอร์ของตัวเอง คือ จากตัวแปร biosbuffer ซึ่งบรรจุข้อความที่ได้จากการเรียกฟังก์ชัน autotype จนกระทั่งข้อความใน biosbuffer จนฟังก์ชัน newkb ก็จะคืนเวกเตอร์อินเตอร์รัพต์เดิมกลับอย่างเก่า

ตัวอย่างการใช้งานคีย์บอร์ดมาโคร

ในรูปที่ 4 โปรแกรม demoauto.c เป็นตัวอย่างโปรแกรมนำเอาโมดูล Autotype.c ให้ใช้งาน ในลักษณะของคีย์บอร์ดมาโครอย่างง่าย โดยเมื่อโปรแกรมจะให้เลือกแป้นพิมพ์ที่จะทำหน้าที่แทนข้อความ (คีย์บอร์ดมาโคร) จากนั้นจะให้พิมพ์ข้อความที่จะถูกแทนที่ แล้วทดสอบโดยให้พิมพ์ข้อความใด ๆ โดยถ้ามีการกดแป้นพิมพ์ที่เป็นคีย์บอร์ดมาโครโปรแกรมก็จะพิมพ์ข้อความที่ถูกแทนออกมาให้

บังคับโปรแกรมอื่นให้ทำตามที่ต้องการ

โปรแกรมรูปที่ 5 เป็นอีกตัวอย่างที่เป็นการนำเอาการพิมพ์อัตโนมัติไปใช้งานเพื่อบังคับโปรแกรมอื่นให้ทำงานตามคำสั่งหรือข้อความที่จัดใส่ล่วงหน้าในโปรแกรม driver.c เป็นการบังคับให้โปรแกรมโลตัสโหลดการไฟล์ข้อมูล TEST.WK1 ขึ้นในเวอร์กชีตโดยอัตโนมัติ

การนำเอาการพิมพ์อัตโนมัติไปใช้งานในลักษณะบังคับโปรแกรมอื่นนั้น

รูปที่ 4
โปรแกรม demoauto.c เป็นตัวอย่างโปรแกรมที่แสดงคีย์บอร์ดมาโครแบบง่าย ๆ (โปรแกรมต้องฉิงค์รวมกับโปรแกรม Autotype.c และ keys.c)

```
/* ------------ demoauto.c ------------- */

/*
 * A program to test autotyping
 */

#include <process.h>
#include <stdio.h>
#include <conio.h>
#include "keys.h"

void main(void)
{
        int mkey, c;
        char macro[80];

        cprintf("\r\nPress a key to use for a macro ... ");
        mkey = getkey();
        cprintf("\r\nEnter the macro string ...\r\n");
        *macro = 78;
        cgets(macro);
        repeat_rate (50);
        cprintf("\r\nNow type some text (Esc when done) ...\r\n"
        while ((c = getkey()) != ESC)
                if (c == mkey)
                        autotype(macro+2);
                else    {
                        putch(c);
                        if (c == '\r')
                                putch('\n');
                }
}
```

รูปที่ 5
โปรแกรม driver.c เป็นตัวอย่างโปรแกรมที่สามารถเรียกโปรแกรมโลตัส 1-2-3 ขึ้นมาทำงานแล้วบังคับให้ โหลดเอาเวอร์กชีต TEST.WK1 เข้ามาเองอัตโนมัติ

```
/* -------- driver.c ------------ */

/*
 * A program to test autotyping into another program
 */

#include <process.h>
#include <stdio.h>
#include <conio.h>
#include "keys.h"

void main(void)
{
        repeat_rate(1000);
        autotype("/frtest.wk1\r");
        system("123");
}
```

อาจจะไม่ได้ผลกับทุกโปรแกรม เพราะบางโปรแกรมอาจจัดการแป้นพิมพ์เองโดยไม่อาศัยไบออสตัวอย่างเช่น โปรแกรมดีเบสทรีพลัส จะจัดการกับแป้นพิมพ์เองโดยมีบัฟเฟอร์ขนาด 15 ตัวอักษรเหมือนไบออส ซึ่งลักษณะนี้ทำให้ไม่สามารถบังคับด้วยการพิมพ์อัตโนมัติได้ หรือโปรแกรมโลตัส 1-2-3 ซึ่งใช้ในโปรแกรมรูปที่ 5 ก็เช่นกันไม่สามารถจะบังคับอัตราเร็วของการพิมพ์ข้อความได้

นอกจากนั้นโปรแกรมประยุกต์บางโปรแกรมมีการเคลียร์บัฟเฟอร์แป้นพิมพ์ทิ้งออกก่อนจะเริ่มทำงานโดยการอ่านแป้นพิมพ์ทิ้ง จนกระทั่งบัฟเฟอร์แป้นพิมพ์ว่าง ซึ่งลักษณะสามารถหลอกโปรแกรมประยุกต์เหล่านั้นได้ ด้วยการใส่ค่า 0×ff ลงไปเป็นตัวอักษรแรกของข้อความที่จะให้อัตโนมัติ ซึ่งฟังก์ชัน newkb จะส่งค่าสถานะแป้นพิมพ์กลับให้โปรแกรมประยุกต์

มุมภาษาซีในตอนนี้ก็ได้เสนอเรื่องราวการปรับแต่งแป้นพิมพ์ ซึ่งส่วนหนึ่งที่เกี่ยวข้องกันกับตอนที่แล้วจบลง ในตอนต่อไปมุมภาษาซีก็จะนำเรื่องอื่นที่น่าสนใจมาเสนอให้ท่านผู้อ่านอีกเช่นเคย พบกันตอนต่อไปครับ